UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/589,239	01/03/2007	Ivan Boule	LSN-4786-8	6941

23117 7590 09/16/2010
NIXON & VANDERHYE, PC
901 NORTH GLEBE ROAD, 11TH FLOOR
ARLINGTON, VA 22203

EXAMINER	
SADLER, NATHAN	

ART UNIT	PAPER NUMBER
2189	

MAIL DATE	DELIVERY MODE
09/16/2010	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)
	10/589,239	BOULE ET AL.
	Examiner	Art Unit
	Nathan Sadler	2189

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *19 July 2010*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-9,11-24,26-38,40-46 and 48-50* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-9,11-24,26-38,40-46 and 48-50* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Continued Examination Under 37 CFR 1.114

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on July 19, 2010 has been entered.

Claim Objections

2. Claim 49 is objected to because of the following informalities: "index a mask associated **wit** said lowest of said levels". Appropriate correction is required.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

4. Claims 1-7, 9, 11, 17-18, 23-24, 26, 35-38, 40, 44-46, and 48 are rejected under 35 U.S.C. 102(b) as being anticipated by McMahon et al. (US 5,784,699).

5. In regards to claim 1, McMahon teaches a method of using a computer having a CPU linked to a digital memory for processing requests for allocation of a memory block

of a data memory, wherein segments of the data memory are associated with different levels according to their size, the method comprising:

6. (a) said CPU receiving a request for the allocation of a memory block within said digital memory (“Dynamic allocation of memory, or dynamic memory, is memory requested by software programs during operation of the software programs.”, Col. 1, lines 13-17) including receiving a binary data set indicative of requested memory block size (“For example, in the C programming language, the C run time library of many systems includes the functions malloc (size) and free(pointer).”, Col. 1, lines 36-39) wherein each bit of the binary data set is associated with an entry of a lookup table associated with one of said levels (“For example, the size based free lists technique may utilize a power of two selection technique to designate 16 byte, 32 byte, 64 byte, 128 byte, 256 byte, etc., blocks as the predetermined block sizes. All requests for memory are rounded to one of these block sizes.”, Col. 2, lines 42-46, this demonstrates that each bit of the binary data set is associated with a predetermined block size, the binary data set is a subset of the bits used to represent the size of the requested memory block ranging from the smallest allowed block to the largest, the lookup table is the master bit map index, 310, 340, figures 3B and 3C);

7. (b) said CPU determining the lowest of said levels containing a segment equal to or larger in size than the requested memory block including determining a most significant set bit of the binary data set (“The size of the requests are rounded up to the nearest slot size.”, abstract, rounding up to the nearest slot size is the same as determining a most significant set bit when the slot sizes are power of two), and

determining from the lookup table entry associated with the most significant set bit a lowest of said levels (“To determine whether a memory block is available in a particular group, the dynamic memory allocator examines the corresponding bit flag in the master bit map index.", Col. 7, lines 30-33) containing a segment of a size equal to or larger than the requested memory block (“For a 112 byte block request, the dynamic memory allocator computes that the free list 7, which indicates the status of 112 byte memory blocks, is part of group 1.”, Col. 7, lines 55-57);

8. (c) said CPU determining, in the level determined in step (b), the availability of a free segment of a size equal to or larger than the requested memory block (step 120, figure 2); and

9. (d) said CPU allocating a free segment (step 135, figure 2) depending on the determination in step (c) (step 130, figure 2).

10. In regards to claim 2, McMahon further teaches e) repeating steps (c) and (d) for a next higher level if no free segment of a size equal to or larger than the requested memory block has been found in step (c) (step 140, figure 2); and

11. (f) repeating step (e) until a free segment has been allocated or there is no next level (the no arrow of step 150, figure 2).

12. In regards to claim 3, McMahon further teaches that each level is associated with a different granule size to the power of two (“For example, the size based free lists technique may utilize a power of two selection technique to designate 16 byte, 32 byte, 64 byte, 128 byte, 256 byte, etc., blocks as the predetermined block sizes.”, Col. 2, lines 42-45), and sizes of memory segments allocated to a level are related to the granule

size of the respective level ("For example, the size based free lists technique may utilize a power of two selection technique to designate 16 byte, 32 byte, 64 byte, 128 byte, 256 byte, etc., blocks as the predetermined block sizes.", Col. 2, lines 42-45).

13. In regards to claim 4, McMahon further teaches that the granule size associated with a level defines a size difference between memory segments allocated to that level ("For example, the size based free lists technique may utilize a power of two selection technique to designate 16 byte, 32 byte, 64 byte, 128 byte, 256 byte, etc., blocks as the predetermined block sizes.", Col. 2, lines 42-45).

14. In regards to claim 5, McMahon further teaches that step (a) further comprises rounding the requested memory block to a lowest granule size before performing steps (b) to (d) ("The block size rounding technique attempts to reduce fragmentation of memory by rounding all requests up to some minimum size.", Col. 2, lines 18-20).

15. In regards to claim 6, McMahon further teaches that each level is associated with a table of pointers indicative of memory addresses of free memory segments of a size allocated to the respective level ("If a memory block is available for the appropriate bin size, then the dynamic memory allocator 50 assigns a memory block utilizing the pointer on the free list as shown in blocks 130 and 135", Col. 5, lines 32-35).

16. In regards to claim 7, McMahon further teaches that step (d) comprises returning a pointer to the allocated free segment ("In response, the dynamic memory allocator 50 returns a pointer to a memory block that fulfills the memory request", Col. 5, lines 1-3).

17. In regards to claim 9, McMahon further teaches that a bitmap is indicative of the state of memory segments (free, allocated), the bitmap comprising a root bitmap, each

bit of the root bitmap being indicative of whether or not an associated one of said levels contains at least one free segment (master bit map index, 340, figure 3C), and wherein step (b) further comprises determining from the root bitmap said lowest level containing a segment of a size equal to or larger than the requested memory block (“From the master bit map index 340, the dynamic memory allocator determines that the next appropriate non-empty group of free lists is group 6.”, Col. 8, lines 2-4).

18. In regards to claim 11, McMahon further teaches that each mask of a set of predetermined masks is associated with one of said levels, and step (c) further comprises performing a logic operation on the mask associated with the lowest level determined in step (b) and said binary data set, wherein a result of said logic operation is an index to bits of the bitmap indicative of the state of a segment of a size equal to or larger than the requested memory block (“Then, the dynamic memory allocator, utilizing the group 1 bit map index 360, masks off the lower bit flags, corresponding to free lists 1-6, to search for a memory block greater than or equal to a 112 byte block. Because bit flags 7-32 are all set to "0", which indicates that there are no available memory blocks in free lists 7-32, the dynamic memory allocator returns to the master bit map index 340.”, Col. 7, line 62 - Col. 8, line 2).

19. In regards to claim 17, McMahon teaches a method of using a computer having a CPU linked to a digital data memory for managing a data memory, the method comprising:

20. said CPU defining a number of levels of the digital data memory (free lists, Col. 4, line 33);

21. said CPU defining a different range of memory segment sizes for each level (bin size, Col. 4, line 49); and

22. said CPU generating a lookup table (master bit map index, 310, 340, figures 3B and 3C), wherein each entry of the lookup table is associated with a bit of a binary data set that indicates size of a requested memory block ("Specifically, each bit flag indicates whether at least one memory block within the corresponding group is available.", Col. 7, lines 19-21, the entry is associated with a bit because the entry is a bit), and wherein each entry of the lookup table indicates one of the levels (the levels are the corresponding groups), whereby a request for allocation of a memory block is processable by determining a level containing segments of a size equal to or larger than the requested memory block using the lookup table ("To determine whether a memory block is available in a particular group, the dynamic memory allocator examines the corresponding bit flag in the master bit map index.", Col. 7, lines 30-33), and allocating a free segment of the same size or larger than the requested memory block in that level (step 135, figure 2).

23. defining a different range of a plurality of different sizes of memory segments for each level, wherein the size of each memory segment is related to the granule size of the respective level ("The rounding factor is defined as the maximum amount of memory a request is rounded to execute a dynamic memory allocation operation.", Col. 4, lines 50-53), and wherein a request for the allocation of a memory block is processable by determining a level containing segments of the same size as or larger than the

requested memory block (step 120, figure 2), and allocating a free segment of a size the same as or larger than the requested memory block in that level (step 135, figure 2).

24. In regards to claim 18, McMahon further teaches that the granule size defines the size difference between memory segments in each level ("The rounding factor is defined as the maximum amount of memory a request is rounded to execute a dynamic memory allocation operation.", Col. 4, lines 50-53).

25. In regards to claim 23, McMahon further teaches generating a table of pointers for each level indicative of memory addresses of free memory segments of a size associated with the respective level ("If a memory block is available for the appropriate bin size, then the dynamic memory allocator 50 assigns a memory block utilizing the pointer on the free list as shown in blocks 130 and 135.", Col. 5, lines 32-35).

26. In regards to claim 24, McMahon further teaches generating a bitmap indicative of a state of each segment (free, allocated) (group bit map indices, 320, 330, 350, and 360, figures 3B and 3C, "In one embodiment, a bit flag set as a '1' or high logic level indicates that the corresponding slot contains an available memory block", Col. 7, lines 28-30) and of whether or not a level contains at least one free segment (master bit map index, 310, 340, figure 3B and 3C "Specifically, each bit flag indicates whether at least one memory block within the corresponding group is available.", Col. 7, lines 19-21),

27. wherein each bit of the third stage bitmaps is associated with an entry in the tables of pointers ("If a memory block is available for the appropriate bin size, then the dynamic memory allocator 50 assigns a memory block utilizing the pointer on the free list as shown in blocks 130 and 135.", Col. 5, lines 32-35, "The bit flags in each group bit

map index indicate availability of a memory block for a corresponding block size.", Col. 7, lines 26-28).

28. In regards to claim 26, McMahon further teaches generating a set of masks, wherein each of the set of masks is associated with one of said levels, and wherein a logical operation of a binary data set indicative of the size of the requested memory block and the mask associated with a level containing segments of the same size as or larger than the requested memory block results in an index to a segment of a size the same as or larger than the requested memory block in that level ("Then, the dynamic memory allocator, utilizing the group 1 bit map index 360, masks off the lower bit flags, corresponding to free lists 1-6, to search for a memory block greater than or equal to a 112 byte block.", Col. 7, lines 62-65).

29. In regards to claim 35, McMahon further teaches allocating free segments of the data memory to different levels according to their size (step 120, figure 2); and

30. providing a bitmap comprising different stages (see figure 3C), wherein the bits of one stage are indicative of the availability of free segments in said levels (master bit map index and arrows, figure 3C), and the bits of another stage are indicative of the state and/or size and/or location of individual segments (group bit map indices, 350 and 360, figure 3C).

31. In regards to claim 36, McMahon further teaches the bits of one stage are associated with pointers indicative of the memory address of free segments ("If a memory block is available for the appropriate bin size, then the dynamic memory allocator 50 assigns a memory block utilizing the pointer on the free list as shown in

blocks 130 and 135.", Col. 5, lines 32-35, "The bit flags in each group bit map index indicate availability of a memory block for a corresponding block size.", Col. 7, lines 26-28).

32. In regards to claim 37, McMahon further teaches updating the bitmap to reflect the allocation or release of memory segments ("a bit map index is maintained to identify available memory blocks", Col. 4, lines 38-39).

33. In regards to claim 38, McMahon further teaches when freeing a memory segment:

34. determining the state of memory segments adjacent to the memory segment to be freed ("To free a large block of memory, adjacent blocks, marked as free memory blocks, are merged together, if possible, through use of the double linked list.", Col. 11, lines 34-36); and

35. merging the memory segment to be freed with free adjacent memory segments ("To free a large block of memory, adjacent blocks, marked as free memory blocks, are merged together, if possible, through use of the double linked list.", Col. 11, lines 34-36).

36. In regards to claim 40, McMahon further teaches an operating system for a computer (operating system, 90, figure 1), adapted to perform the method of claim 1.

37. In regards to claim 44, McMahon further teaches a computer program adapted to perform the method of claim 1 when operated on a computer ("The dynamic memory allocation system may be implemented in either hardware or software.", Col. 16, lines 13-14).

38. In regards to claim 45, McMahon further teaches a storage medium having stored thereon a set of instructions, which when executed by a computer, performs the method of claim 1 (“Prior to loading into a general purpose computer system, the dynamic memory allocation system software may reside as encoded information on a computer readable medium", Col. 16, lines 17-20).

39. In regards to claim 46, McMahon further teaches a computer system comprising the operating system of claim 40 (“Prior to loading into a general purpose computer system, the dynamic memory allocation system software may reside as encoded information on a computer readable medium", Col. 16, lines 17-20).

40. In regards to claim 48, McMahon further teaches defining a different granule size for each level, wherein the size of each memory segment is related to the granule size of the respective level (“For example, the size based free lists technique may utilize a power of two selection technique to designate 16 byte, 32 byte, 64 byte, 128 byte, 256 byte, etc., blocks as the predetermined block sizes.”, Col. 2, lines 42-46).

Claim Rejections - 35 USC § 103

41. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

42. Claim 8 is rejected under 35 U.S.C. 103(a) as being unpatentable over McMahon et al. (US 5,784,699) in view of RedHat ("MALLOC(3)").

43. McMahon teaches the method of claim 1 as discussed above. McMahon fails to teach returning a null pointer if no free segment is allocated. RedHat teaches returning a null pointer if no free segment is allocated ("NULL if the request fails", RETURN VALUES). It would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine McMahon and RedHat to return a null pointer if no free segment is allocated because RedHat describes the function of malloc() which McMahon discusses (Col. 12, lines 52-54).

44. Claims 12-16 and 19-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over McMahon et al. (US 5,784,699) in view of Morzy et al. ("Hierarchical Bitmap Index: An Efficient and Scalable Indexing Technique for Set-Valued Attributes").

45. In regards to claim 12, McMahon further teaches that said bitmap comprises a plurality of second stage bitmaps (group bit map indices, 350 and 360, figure 3C), each bit of the root bitmap being indicative of the state of the bits of an associated one of said second stage bitmaps (bits 1 and 6 of master bit map index, figure 3C), and each bit of the second stage bitmap being indicative of whether or not an associated segment is free ("The bit flags in each group bit map index indicate availability of a memory block for a corresponding block size", Col. 7, lines 26-28), and wherein the operation result is an index to one bit of the second stage bitmap and said one bit of the second stage bitmap being indicative of the state of a segment of a size the same as or larger than the requested memory block ("In one embodiment, a bit flag set as a "1" or high logic level indicates that the corresponding slot contains an available memory block", Col. 7,

lines 28-30). McMahon fail to teach a plurality of third stage bitmaps and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps. Morzy teaches a plurality of third stage bitmaps (index key leaves, figure 1) and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps ("At the upper level (level 2) 4 bits representing non-empty leaf nodes are set to '1'.", page 243, paragraph 1) in order to scalably and efficiently index large collections (page 250, paragraph 4). It would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine McMahon and Morzy for a plurality of third stage bitmaps and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps in order to scalably and efficiently index large collections (Morzy, page 250, paragraph 4).

46. In regards to claim 13, McMahon further teaches that if no free segment is found, repeating the determination for the next more significant bit of said predetermined number of bits of the third stage bitmap, until a free segment is found or there is no more significant bit of said predetermined number of bits of the third stage bitmap (step 145, figure 2).

47. In regards to claim 14, McMahon further teaches that if no free segment is found, repeating the determination for the next more significant bit of said predetermined number of bits of the third stage bitmap, until a free segment is found or there is no

more significant bit of said predetermined number of bits of the third stage bitmap (yes arrow of step 150, figure 2).

48. In regards to claim 15, McMahon further teaches that if no free segment is found, repeating the determination for the second stage bitmap associated with the next more significant set bit of the root bitmap, until a free segment is found or there is no more significant bit of the root bitmap (no arrow of step 150, figure 2).

49. In regards to claim 16, McMahon further teaches that each bit of the third stage bitmaps is associated with an entry in a table of pointers indicative of memory addresses of free memory segments ("If a memory block is available for the appropriate bin size, then the dynamic memory allocator 50 assigns a memory block utilizing the pointer on the free list as shown in blocks 130 and 135", Col. 5, lines 32-35).

50. In regards to claim 19, McMahon further teaches that the bitmap comprises a root bitmap (310, figure 3B), each level being associated with one bit of the root-bitmap (see arrows, figure 3B), and a plurality of second stage bitmaps associated with the segments (groups, 315-330, figure 3B), each bit of the root bitmap being indicative of the state of the bits of an associated one of said second stage bitmaps ("The bit flags in each group bit map index indicate availability of a memory block for a corresponding block size.", Col. 7, lines 26-30). McMahon fails to teach a plurality of third stage bitmaps and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps. Morzy teaches a plurality of third stage bitmaps (index key leaves, figure 1) and each bit of said second stage bitmaps being indicative of the state of an associated predetermined

number of bits of one of said third stage bitmaps (“At the upper level (level 2) 4 bits representing non-empty leaf nodes are set to ‘1’.”, page 243, paragraph 1) in order to scalably and efficiently index large collections (page 250, paragraph 4). It would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine McMahon with Morzy for a plurality of third stage bitmaps and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps in order to scalably and efficiently index large collections (Morzy, page 250, paragraph 4).

51. In regards to claim 20, Morzy further teaches that the bitmap comprises a root bitmap (index key root level, figure 1), each level being associated with one bit of the root-bitmap (arrows from index key root level, figure 1), and a plurality of second (inner nodes, figure 1) and third stage bitmaps (index key leaves, figure 1) associated with the segments, each bit of the root bitmap being indicative of the state of the bits of an associated one of said second stage bitmaps (“In the root of the index key only first and third bits are set to ‘1‘, which means that only first and third inner nodes at the level 2 are non-empty.”, page 243, paragraph 1), and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps (“At the upper level (level 2) 4 bits representing non-empty leaf nodes are set to ‘1’.”, page 243, paragraph 1).

52. In regards to claim 21, McMahon further teaches updating the bitmap when a segment is allocated (“a bit map index is maintained to identify available memory blocks”, Col. 4, lines 38-39).

53. In regards to claim 22, McMahon further teaches updating the bitmap when a segment is freed ("a bit map index is maintained to identify available memory blocks", Col. 4, lines 38-39).

54. Claims 27-34 are rejected under 35 U.S.C. 103(a) as being unpatentable over McMahon et al. (US 5,784,699) in view of Wilson et al. ("Dynamic Storage Allocation: A Survey and Critical Review").

55. In regards to claim 27, McMahon teaches a method of claim 17 as discussed above. McMahon further teaches creating a first doubly linked list of consecutive memory segments irrespective of size and status (free, allocated) ("To free a large block of memory, adjacent blocks, marked as free memory blocks, are merged together, if possible, through the use of the double linked list.", Col. 11, lines 34-36). McMahon fails to teach creating a second doubly linked list of free memory segments of the same size.

56. Wilson teaches creating a first doubly linked list of consecutive memory segments irrespective of size and status (free, allocated) ("Many allocators that support general coalescing are implemented using boundary tags (due to Knuth [Knu73]) to support the coalescing of free areas. Each block of memory has a both header and a "footer" field, both of which record the size of the block and whether it is in use.", page 39, paragraph 7) and

57. creating a second doubly linked list of free memory segments ("Many systems use doubly-linked linear lists", page 40, paragraph 4) of the same size ("One of the

simplest allocators uses an array of free lists, where each list holds free blocks of a particular size”, page 51, paragraph 6)

58. in order to “support the coalescing of free areas” (page 39, paragraph 7).

59. It would have been obvious at the time the invention was made to a person of ordinary skill in the art to combine McMahon with Wilson by creating a first doubly linked list of consecutive memory segments irrespective of size and status (free, allocated) and

60. creating a second doubly linked list of free memory segments of the same size

61. in order to “support the coalescing of free areas” (id).

62. In regards to claim 28, Wilson further teaches that memory segments in the first doubly linked list are arranged in the order of associated memory addresses (“Many allocators that support general coalescing are implemented using boundary tags (due to Knuth [Knu73]) to support the coalescing of free areas. Each block of memory has a both header and a ‘footer’ field, both of which record the size of the block and whether it is in use.”, page 39, paragraph 7).

63. In regards to claim 29, Wilson further teaches determining the state of memory segments adjacent to the memory segment to be freed using the first doubly linked list (“Many allocators that support general coalescing are implemented using boundary tags (due to Knuth [Knu73]) to support the coalescing of free areas., page 39, paragraph 7);

64. merging the memory segment to be freed with free adjacent memory segments (“Many allocators that support general coalescing are implemented using boundary tags

(due to Knuth [Knu73]) to support the coalescing of free areas.", page 39, paragraph 7); and

65. updating the first and second doubly linked lists accordingly ("Adjacent free areas are merged to form larger free blocks", page 39, paragraph 7).

66. In regards to claim 30, Wilson further teaches that each second doubly linked list is a LIFO (Last In First Out) list ("it appears that address-ordered first fit may cause significantly less fragmentation than LIFO-ordered first fit", page 44, paragraph 6).

67. In regards to claim 31, Wilson further teaches updating the second doubly linked list upon allocation of a memory segment upon request ("When a request is serviced, the free list for the appropriate size is used to satisfy the request.", page 51, paragraph 6).

68. In regards to claim 32, Wilson further teaches allocating a portion of the determined segment large enough to satisfy the request ("the free list for the corresponding size class is searched for a block at least large enough to hold it.", page 53, paragraph 3);

69. providing the remaining portion as a new free memory segment ("An allocated block", page 1, paragraph 4); and

70. updating the first and second doubly linked lists accordingly ("Each block of memory has a both header and a "footer" field, both of which record the size of the block and whether it is in use.", page 39, paragraph 7, "One of the simplest allocators uses an array of free lists, where each list holds free blocks of a particular size", page 51, paragraph 6).

71. In regards to claim 33, Wilson further teaches that each segment is associated with a header, thereby to form the first doubly linked list, each header including information indicative of the size of the associated segment, information indicative of the state (free, allocated) of the associated segment, and a pointer indicative of the memory address of the previous segment (“Many allocators that support general coalescing are implemented using boundary tags (due to Knuth [Knu73]) to support the coalescing of free areas. Each block of memory has a both header and a "footer" field, both of which record the size of the block and whether it is in use.”, page 39, paragraph 7).

72. In regards to claim 34, Wilson further teaches that the header associated with each free segment of a given size further includes a pointer indicative of the memory address of a previous and/or subsequent free segment of the same size, depending on the availability of a previous and/or subsequent free segment of the same size and in accordance with the order of free segments of the same size in the LIFO list (“For allocators using free lists or indexing trees to keep track of free blocks, the list or tree nodes are generally embedded in the free blocks themselves.”, page 40, paragraph 3).

73. Claim 41 is rejected under 35 U.S.C. 103(a) as being unpatentable over McMahon et al. (US 5,784,699) in view of Stallings (*Operating Systems: Internals and Design Principles*).

74. In regards to claim 41, McMahon teaches the operating system of claim 40. McMahon fails to teach that the operating system is a realtime operating system. Stallings teaches that the operating system is a realtime operating system (“The

operating system, and in particular the scheduler, is perhaps the most important component of a real-time system.", page 450, paragraph 1) in order to "control or react to events that take place in the outside world" (page 450, paragraph 2). It would have been obvious at the time the invention was made to a person of ordinary skill in the art to combine McMahon with Stallings such that the operating system is a realtime operating system in order to "control or react to events that take place in the outside world" (id).

75. Claims 42-43 are rejected under 35 U.S.C. 103(a) as being unpatentable over McMahon et al. (US 5,784,699) in view of Rubini et al. (*Linux Device Drivers*).

76. In regards to claim 42, McMahon teaches the operating system of claim 40. McMahon fails to teach performing the method described above at task level. Rubini teaches performing memory allocation at task level ("GFP_USER Used to allocate memory on behalf of the user. It may sleep, and is a low-priority request.", section 7.1.1) because it may sleep and is a low-priority request (section 7.1.1). It would have been obvious at the time the invention was made to a person of ordinary skill in the art to combine McMahon with Rubini by performing the method described above at task level because it may sleep and is a low-priority request (id).

77. In regards to claim 43, McMahon teaches the operating system of claim 40. McMahon fails to teach performing the method described above at interrupt level. Rubini teaches performing memory allocation at interrupt level ("GFP_ATOMIC Used to allocate memory from interrupt handlers and other code outside of a process context.

Never sleeps.", section 7.1.1) because it never sleeps (section 7.1.1). It would have been obvious at the time the invention was made to a person of ordinary skill in the art to combine McMahon with Rubini by performing the method described above at interrupt level because it never sleeps (section 7.1.1).

78. Claims 49 and 50 are rejected under 35 U.S.C. 103(a) as being unpatentable over Masmano et al. ("Dynamic storage allocation for real-time embedded systems") in view of Moreno ("A Wrapper for Look-Up Tables (LUT) Operations in C++").

79. In regards to claim 49, Masmano teaches a method of using a computer having a CPU linked to a digital data memory for processing requests for allocation of a memory block of a data memory, wherein segments of the data memory are associated with different levels according to segment size, the method comprising:

80. said CPU receiving a request for the allocation of a memory block within said digital data memory including receiving a binary data set indicative of requested memory block size (size, section 6, paragraph 2) wherein each bit of the binary data set is associated with one of said levels ("The first-level array divides free blocks in classes that are a power of two apart", section 5, paragraph 1);

81. said CPU determining the lowest of said levels containing a segment equal to or larger in size than the requested memory block including determining a most significant set bit of the binary data set (first level index, section 6, paragraph 2), and determining based on the most significant set bit a lowest of said levels containing a segment of a

size equal to or larger than the requested memory block ("search the next (bigger than the requested size) non empty list in the TSLF structure", section 6, paragraph 7);

82. providing a bitmap indicative of a state of memory segments (free, allocated) (bitmap associated with second-level, section 5, paragraph 1);

83. providing a plurality of masks, wherein each mask is associated with one of said levels ("The second level index can be obtained using a mask on the first SLI bits of the size that follow the last bit set.", section 6, paragraph 2);

84. said CPU using the most significant set bit to index a mask associated wit said lowest of said levels containing a segment of a size equal to or larger than the requested memory block ("The second level index can be obtained using a mask on the first SLI bits of the size that follow the last bit set.", section 6, paragraph 2);

85. said CPU logically combining said indexed mask with said binary data set to output an index to said bitmap ("The second level index can be obtained using a mask on the first SLI bits of the size that follow the last bit set.", section 6, paragraph 2); and

86. said CPU determining, from said bitmap, the availability of a free segment of a size equal to or larger than the requested memory block using said index to said bitmap ("search the next (bigger than the requested size) non empty list in the TSLF structure. … If a list is found, hen the block at the head of the list will be used to fulfil the request.", section 6, paragraph 7).

87. Masmano fails to teach a lookup table with entries associated with each of said levels used to index a mask based on the most significant set bit; and that the plurality of masks are predetermined. Moreno teaches using a lookup table to index

predetermined values (“Look-Up Tables are a technique commonly used to accelerate numeric processing in applications with demanding timing requirements.”, “The basic idea is to pre-compute the result of complex operations that are or can be expressed as a function of an integer value. The pre-computed results are typically stored in an array, which is used at run-time instead of performing the whole, time-consuming operation.”, Introduction, paragraph 1 and 2) in order to reduce the overhead at run-time (id). It would have been obvious at the time the invention was made to a person of ordinary skill in the art to combine Masmano with Moreno to include a lookup table with entries associated with each of said levels used to index a mask based on the most significant set bit; and that the plurality of masks are predetermined in order to reduce the overhead at run-time (id).

88. In regards to claim 50, Masmano teaches a method of using a computer having a CPU linked to a digital data memory for managing a data memory, the method comprising:

89. said CPU defining a number of levels of the digital data memory (“In order to speedup the access to the free blocks and also to manage a large set of segregated lists (to reduce fragmentation) the array or lists has been organized as a two level array.”, section 5, paragraph 1);

90. said CPU defining a different range of memory segment sizes for each level (“The first-level array divides free blocks in classes that are a power of two apart (8, 16, 32, 64, etc.)”, section 5, paragraph 1); and

91. said CPU generating masks, wherein each mast is associated with a bit of a binary data set that indicates size of a requested memory block, and wherein each mask indicates one of the levels (“The second level index can be obtained using a mask on the first SLI bits of the size that follow the last bit set.”, section 6, paragraph 2), whereby a request for allocation of a memory block is processable by determining a level containing segments of a size equal to or larger than the requested memory block using masks, and allocating a free segment of the same size or larger than the requested memory block in that level (“search the next (bigger than the requested size) non empty list in the TSLF structure”, section 6, paragraph 7), said method including:

92. (a) said CPU receiving a request for the allocation of a memory block within said digital data memory including receiving a binary data set indicative of requested memory block size (size, section 6, paragraph 2) wherein each bit of the binary data set is associated with a mask associated with one of said levels (“The first-level array divides free blocks in classes that are a power of two apart (8, 16, 32, 64, etc.)”, section 5, paragraph 1);

93. (b) said CPU determining the lowest of said levels containing a segment equal to or larger in size than the requested memory block including determining a most significant set bit of the binary data set (“The first level index ... can be computed as the position of the last bit set (bit set to one) of the size.", section 6, paragraph 2), and determining from the mask associated with the most significant set bit a lowest of said levels containing a segment of a size equal to or larger than the requested memory

block ("search the next (bigger than the requested size) non empty list in the TSLF structure", section 6, paragraph 7);

94. (c) said CPU determining, in the level determined in step (b), the availability of a free segment of a size equal to or larger than the requested memory block ("If a list is found, then the block at the head of the list will be used to fulfil the request.", section 6, paragraph 7);

95. (d) said CPU allocating a free segment depending on the determination in step (c) ("If a list is found, then the block at the head of the list will be used to fulfil the request.", section 6, paragraph 7); and

96. (e) said CPU determining, from said bitmap, the availability of a free segment of a size equal to or larger than the requested memory block using said index to said bitmap ("Each array of lists has an associate bitmap used to mark which lists are empty and which ones contain free blocks.", section 5, paragraph 1).

97. Masmano fails to teach said CPU generating a lookup table wherein each entry of the lookup table is associated with a mask. Moreno teaches generating a lookup table wherein each entry of the lookup table is associated with a predetermined values ("Look-Up Tables are a technique commonly used to accelerate numeric processing in applications with demanding timing requirements.", "The basic idea is to pre-compute the result of complex operations that are or can be expressed as a function of an integer value. The pre-computed results are typically stored in an array, which is used at run-time instead of performing the whole, time-consuming operation.", Introduction, paragraph 1 and 2) in order to reduce the overhead at run-time (id). It would have been

obvious at the time the invention was made to a person of ordinary skill in the art to combine Masmano with Moreno such that said CPU generates a lookup table wherein each entry of the lookup table is associated with a mask in order to reduce the overhead at run-time (id).

Response to Arguments

98. Applicant's arguments filed July 19, 2010 have been fully considered but they are not persuasive.

99. In response to arguments A and B, the arguments appear to hinge on the lookup table being used regardless of whether the level contains any free segments. However, the lookup table being used **regardless** is not found in the claim limitations. As the Applicants correctly note, “Thus, it can be either." The claims do not specify that it must be both.

100. In response to argument C, as the Applicants have kindly pointed out, at least in some cases, it is done based on the most significant bit. The fact that McMahon does it in other ways in other cases does not prevent McMahon from anticipating the claims. The claims use the transitional phrase “comprising” and are thus open-ended.

101. In response to argument D, McMahon does teach an association between the entries of the master bit index and the bits of a binary data set indicating the size of a requested segment (Table 1 shows that each free list contains blocks of a specific size. Col. 5, lines 22-39 teach that the allocator uses the size of the request to determine which free list to access.)

102. In response to argument F, the arguments are moot in view of the new grounds of rejection.

Conclusion

103. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Nathan Sadler whose telephone number is (571)270-7699. The examiner can normally be reached on Monday - Thursday 8:30-6:00 EST.

104. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Reginald Bragdon can be reached on (571)272-4204. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

105. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/N. S./
Nathan Sadler
Examiner, Art Unit 2189
September 14, 2010

/Reginald G. Bragdon/
Supervisory Patent Examiner, Art Unit 2189